**SW Quality Assurance & Testing: Continuous Web Integration and Deployment for BMI Application**

# Project Overview

Author: Cassie Javorsky

NetID: clj500

GitHub Username: clj500

GitHub Repository: [clj500/BMI-Calculator (github.com)](https://github.com/clj500/BMI-Calculator)

This document contains all planning, development, and testing details for a Body Mass Index calculator application and web interface. The resulting application prompts users to enter their height and weight in inches and pounds, and then calculates and returns the user’s Body Mass Index number based on those measurements. After completion of the initial application, a web interface was developed through \_\_\_\_\_\_\_\_\_. The project documents the continuous integration and unit testing of the application and its corresponding web interface.

# Setup and Execution Instructions

## Downloading and Running the Python Code and Pytests

### Download Project Files

All project documents and python files can be found in the GitHub repository located here:

[clj500/assignment2 (github.com)](https://github.com/clj500/assignment2/tree/main)

These files should all be saved to the same location on your computer.

### Download Python 3.12

To run the BMI program, the user should first download Python 3.12 for Windows 10 from the following page: [Download Python | Python.org](https://www.python.org/downloads/)

The following site provides more in-depth instruction for downloading Python 3.12 and offers multiple download options for Windows and Mac:

[Python 3 Installation & Setup Guide – Real Python](https://realpython.com/installing-python/)

### Download Pytest

To install Pytest for Windows, open the command prompt and type the following command:

pip install pytest

To check that Pytest was successfully downloaded type:

pytest --version

This blog contains more help if this process does not work for you:

[How to Install pytest in Python? – Be on the Right Side of Change (finxter.com)](https://blog.finxter.com/how-to-install-pytest-in-python/#:~:text=How%20to%20Install%20pytest%20on%20Windows%3F%201%20Type,Wait%20for%20the%20installation%20to%20terminate%20successfully.%20)

### Run the BMI Test File

To run the test file (titled “**test\_BMI.py**”) open your command prompt and change directories (cd) into the same location as where you saved the python files. Make sure there are no other python test files in this location and then type “pytest” into the command line. This should run the test file.

A screenshot of a computer program

Description automatically generated

### Run the BMI Program File

To run the program file (titled “**bmiProgram.py**”), right-click on the file and select “Edit with…” and choose IDLE 3.12. Once you have opened the file in IDLE click the run tab and run the program.

A screenshot of a computer

Description automatically generated

## Downloading Django and Running the Web Interface

### Setup Virtual Environment

Before installing Django, a virtual environment should be set up. Open the command prompt and cd into the directory where you would like to install Django (the directory which will contain the website). Then type the following command:

**python -m venv virtualenv**

This command will not show any output in the command line but should create a new folder titled “tutorial-env” within the folder you are working in.

Once you have run this command, make sure you are in the same folder you started in, and run the command:

**virtualenv\Scripts\activate**

This should activate the virtual environment and should show the name of the venv in front of the directory as shown below:

A computer screen with white text

Description automatically generated

### Download Django

To download an official version of Django, cd into the directory where you would like to install Django and activate the virtual environment (this should already be done if you just completed the Setup Virtual Environment step). Once you are in the correct directory run the following command:

**python -m pip install Django**

A screenshot of a computer program

Description automatically generated

# Part 1: Test-Driven Development for BMI Application

## Functions & Respective Unit Tests

Function 1: Input User Height

Function prints a statement: “Enter height (in inches):” which prompts the user to enter their height in inches. This value is stored as a float called height for later conversion and use in the BMI equation.

Function 2: Input User Weight

Function prints a statement: “Enter weight (in lbs):” which prompts the user to enter their weight in pounds. This value is stored as a float called weight for later conversion and use in the BMI equation.

Function 3: Convert Height

Function converts any height from inches to centimeters to plug into the BMI equation. The conversion happens by multiplying the height in inches by 0.025 and then squaring that answer. The test (ad-hoc) for this function checks that the conversion outputs the correct corresponding value:

A screenshot of a math problem

Description automatically generated

Function 4: Convert Weight

Function converts any weight from pounds to kilograms to plug into the BMI equation. The conversion happens by multiplying the weight in pounds by 0.45. The test (ad-hoc) for this function checks that the conversion outputs the correct corresponding value:

A white background with black text

Description automatically generated

Function 5: Calculate BMI

Function takes height and weight in the parameters and divides the weight by the height to calculate a corresponding BMI value. The test (ad-hoc) for this function checks that the calculation outputs the right result.

A white background with black numbers and red text

Description automatically generated

Function 6: Identify BMI Category

A screenshot of a computer code

Description automatically generatedA screenshot of a computer code

Description automatically generatedFunction takes BMI in the parameter and uses an if-else condition statement to determine which of four categories the BMI value falls under. There are four separate tests for this function: one for each category. Each test verifies that the boundaries for each category are established correctly in the if-else condition statement through Weak N x 1 boundary testing strategies.

## Boundary Testing Techniques & Boundary Shift

This project utilized a Weak N x 1 boundary testing technique. A Weak N x 1 boundary test is ideal for identifying the maximum number of boundary issues, and can easily catch boundary shift problems.

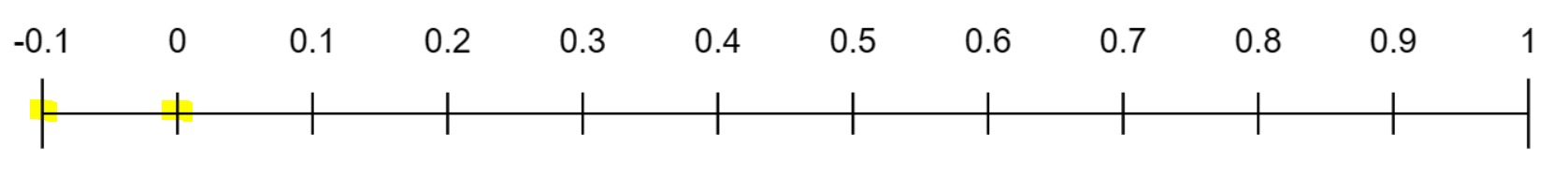
The categories for BMI were provided in the following format:

|  |  |
| --- | --- |
| **BMI** | **Category** |
| <18.5 | Underweight |
| 18.5-24.9 | Normal Weight |
| 25-29.9 | Overweight |
| >=30 | Obese |

This information was interpreted and translated to the following format:

|  |  |
| --- | --- |
| **BMI** | **Category** |
| [0, 18.5) | Underweight |
| [18.5, 25) | Normal Weight |
| [25, 30) | Overweight |
| [30, ∞] | Obese |

To test the Underweight boundary, testing took place at the following points:



A black line with yellow circle

Description automatically generated

-0.1 = OFF

0 = ON

18.4 = interior

18.5 = ON

18.6 = OFF

To test the Normal Weight boundary, testing took place at the following points:

A black line with a yellow circle

Description automatically generated

A black line with a yellow dot

Description automatically generated

18.4 = OFF

18.5 = ON

18.6 = interior

24.9 = OFF

25 = ON

To test the Overweight boundary, testing took place at the following points:

A number on a white background

Description automatically generated

A black and white image of a number

Description automatically generated

24.9 = OFF

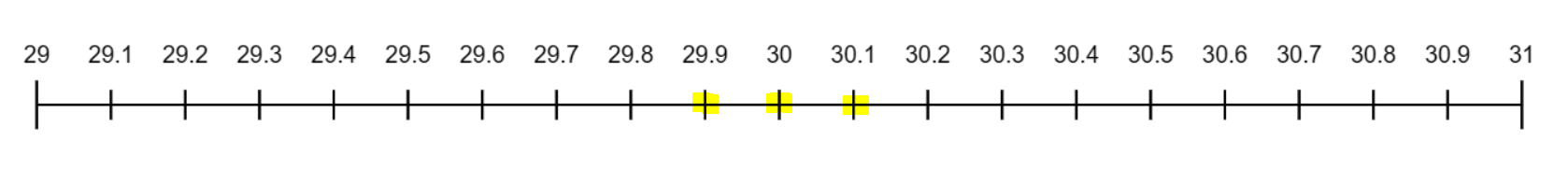
25 = ON

25.5 = interior

29.9 = OFF

30 = ON

To test the Obese boundary, testing took place at the following points:



29.9 = OFF

30 = ON

30.1 = OFF/interior

A screenshot of a computer program

Description automatically generated

After applying a boundary shift of 0.1 to the lower boundary of the Normal Weight category:

A screenshot of a computer code

Description automatically generated

A computer screen shot of a program

Description automatically generated

Two tests failed because their value was now completely excluded from any boundaries. They caught this because they tested exactly on the boundary.

# Part 2: Continuous Web Integration and Deployment for BMI Application

## Tool Description

### Django: Python-Based Web Framework

The project utilizes Django, a python-based web framework, to implement the web interface for the BMI application. Since the original script for the BMI program was written in Python, Django was the ideal choice for web development as it allows seamless integration of Python code into a website. Without Django, this process would require more time and external tools and integrations.

### CircleCI: Continuous Integration and Development Program

To allow for continuous integration throughout the project, CircleCI is utilized by developers to monitor, maintain, and communicate about updates to the project, including coding and deployment.

Deployment Pipeline

## Cloud Platform Usage

## Code Coverage Report